

REMARKS

This Amendment After Final is filed in response to the outstanding final Office Action dated April 17, 2007. A Petition for Extension of Time accompanies the Amendment After Final, extending the time for applicants' response, up to and including August 17, 2007. Claims 1, 7, 15, and 21 are amended hereby; claims 2, 3, 8, 10, 16, 20 and 22 are cancelled without prejudice or disclaimer of subject matter. Claims 1, 4-7, 11, 12, 15, 17-19 and 21 remain pending hereinafter, where claims 1, 15 and 21 are the independent claims.

At paragraph 7 of the outstanding final Office Action, the Abstract was objected to with reference to MPEP 608.01(b). In response, applicants have amended the Abstract, shown at page 2 of this paper, and respectfully assert that the Abstract as amended is now in proper form.

At paragraph 9 of the outstanding final Office Action, claims 10-12 were objected to under 37 CFR 1.75(c), where the Examiner asserts that the rejected claims are of improper dependent form for not referring to a preceding claim. In response, applicants assert that each of claims 11-12 (applicant has cancelled claim 10, incorporating the substance of same claim into claim 21) are dependent from claim 21, which dependencies were changed from claim 9 to claim 21 in the Amendment filed on February 21, 2007, in this case. 37 CFR 1.75(c), sets forth in pertinent part, that "[o]ne or more claims may be presented in dependent form, referring back to and further limiting another claim or claims in the same application." In applicants' understanding, the dependent claims may refer to a claim from which they depend that does not "precede" the dependent claims (have a great claim number) under the cited rule, as in the instant

case with respect to dependent claims 10-12. Applicants, therefore, request withdrawal of the objection to pending claims 11 and 12.

At paragraphs 10 and 11 of the outstanding final Office Action, claims 1-8, 10-12 and 15-22 were rejected under 35 USC §102(e) as anticipated by US Patent No. 6,721,941 to Morshed, et al. (Morshed).

With respect to independent claims 1 and 15, the Examiner asserts that Morshed discloses program storage device and a software tool containing machine readable instructions stored on a physical medium for monitoring the behavior of a running computer program for code patterns that violate a given set of coding rules (Figs. 2-4 and related text), the software tool comprising: a pattern detector manager including machine readable instructions for inserting into a running computer program a plurality of entry breakpoints (e.g., Fig. 14, blocks 442, 448, 452, 456, 460, 464, col. 23, line 1, to col. 24, line 11), each of said entry breakpoints being associated with one of a plurality of defined coding patterns (e.g., Fig. 14, blocks 446, 450, 454, 458, 462); and a plurality of pattern detectors, each of the pattern detectors being associated with one of said defined coding patterns, including machine readable instructions, and being invoked by the pattern detector manager, after one of the entry breakpoints associated with the coding pattern associated with said each of the pattern detectors, is reached in the computer program (e.g., Fig. 15, col. 24, lines 11-62, Fig. 16, col. 25, lines 1-37), for determining whether the computer program violates the coding pattern associated with said each of the pattern detectors (e.g., Fig. 12, col. 21, lines 6-67 and col. 23, line 36, through col. 24, line 11).

Applicants have amended each of the independent claims of the invention to include various limitations of the cancelled dependent claims in order to clearly distinguish Morshed, so respectfully disagree that claims 1, 15 and 21, as amended, are unpatentable in view of Morshed under section 102. Applicants' invention broadly relates to the automatic detection of problematic coding patterns and violations of best practices patterns at program runtime. As discussed in detail in the present application, in any large software deployment, subtle coding defects can cause problems in successful deployment of the software. Tracking down these defects may be extremely difficult in the production environment because of a number of factors.

One factor is that, in many cases, the symptoms are usually not unique to a particular type of software defect. This makes correlating symptoms to specific defects nearly impossible. Another factor is that many symptoms may manifest themselves only during production level loads and production configurations. This means that the defects are often undetected in the testing and debugging of software. In addition, the reasons why a piece of code is defective are often complex and may span third party libraries and frameworks.

The present invention effectively addresses these issues. Generally, this is done by observing the behavior of a running program within the context of a large number of defined coding patterns, and automatically flagging violations of the coding patterns when they occur. More specifically, in accordance with one aspect of the invention, a software tool is provided for monitoring the behavior of a running computer program for code patterns that violate a given set of coding rules. This software tool comprises a pattern detector manager and a plurality of individual pattern detectors.

The pattern detector manager is provided for inserting into the running computer program a plurality of entry breakpoints, each of which is associated with one of a plurality of defined coding patterns. Each of the pattern detectors is also associated with one of the defined coding patterns, and each of the pattern detectors is invoked by the pattern detector manager, after one of the entry breakpoints associated with the coding patterns that, in turn, is associated with the pattern detector, is reached in the computer program. When invoked, a pattern detector determines whether the computer program violates the coding pattern associated with the pattern detector.

The prior art does not disclose or suggest the use of plural pattern detectors in the manner described above. For example, Morshed discloses a method and computer program for monitoring execution of an application in a computing system, including a debugging procedure whereby debugging information is reported to a program monitor using an application debugging interface. Program execution information is gathered using the debugging information. Morshed's Figs. 2-4, and related text describe a compiler operating with IR code instrumentation, interaction with various stages of the compiler and IR code, and detailed operation of the software for IR instrumentation. Morshed's Fig. 14 and related text describe instrumenting a method of a class, Fig. 12 and related text describe the method instrumenting byte code, Figs 15-17 and related text describe portions of Fig. 14, and its class operation in greater detail.

As distinguished from Morshed, the focus of the present invention is the use of a debugger to automatically detect code patterns that violate a given set of coding rules. An

example of a coding rule is: an invocation of method A should never follow an invocation of method B in a calling sequence. Such rules are generally impossible to verify prior to program runtime, because a compiler may not be able to enumerate all possible control flow paths through a static analysis of the program code. It is also impractical to enforce such a coding rule as a programming discipline. This invention provides a novel way of using a debugger (a tool that is conventionally used to determine the cause of a program bug, as in Stall et al.'s work) as an automatic verification tool to check when such coding rules are being violated, and to show the programmer the flow of control that created the violation.

Unlike debugging techniques that rely on manual insertion of breakpoints to pause execution of a running program in order to inspect some state, the present invention uses automatic breakpoint insertion to perform automatic checking of coding rules on the fly while the program is executing. The technique of this invention thus differs substantially from prior art that uses debugging techniques such as multiple breakpoint insertion for manual inspection of state for purposes of understanding the source of a program bug (which is described in Stall et al.'s patent).

Independent Claims 1, 15 and 21 describe important features not shown in or suggested by the prior art, and in particular, these claims describe the use of the pattern detector manager and the plurality of pattern detectors. The pattern detector manager is described in these claims as being used for inserting into the running computer program a plurality of entry breakpoints, each of which is associated with one of a plurality of defined coding patterns. Also, as described in Claims 1, 15 and 21, each of the pattern detectors is associated with one of the defined coding

patterns, and each of the pattern detectors is invoked by the pattern detector manager, after one of the entry breakpoints associated with the coding patterns that, in turn, is associated with that pattern detector, is reached in the computer program. These claims positively describe the further feature that, when invoked, a pattern detector determines whether the computer program violates the coding pattern associated with the pattern detector.

Independent claim 1, as amended hereby, recites a software tool containing machine readable instructions stored on a physical medium for monitoring the behavior of a running computer program for code patterns that violate a given set of coding rules. The software tool includes a pattern detector manager including machine readable instructions for inserting into a running computer program a plurality of entry breakpoints, automatically, with little or no intervention for a user, each of said entry breakpoints being associated with one of a plurality of defined coding patterns and a plurality of pattern detectors, each of the pattern detectors being associated with one of said defined coding patterns, including machine readable instructions, and being invoked by the pattern detector manager, after one of the entry breakpoints associated with the coding pattern associated with said each of the pattern detectors, is reached in the computer program, for determining whether the computer program violates the coding pattern associated with said each of the pattern detectors by inserting into the program at least one further breakpoint for identifying a respective step in the program that is part of the coding pattern associated with said one of the entry breakpoints. The plurality of defined coding patterns is selected from a group comprising best practice patterns and problematic coding patterns.

Independent claim 15, as amended hereby, sets forth a program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for monitoring the behavior of a running computer program. The method includes using a pattern detector manager to insert into a running computer program a plurality of entry breakpoints, each of said entry breakpoints being associated with one of a plurality of defined coding patterns, and monitoring to detect the occurrences of the entry breakpoints in the computer program, and upon detection of one of the entry breakpoints in the computer program, further inserting into the program at least one further breakpoint for identifying a respective step in the program that is part of the coding pattern associated with said one of the entry breakpoints and using a plurality of pattern detectors for monitoring the computer program, wherein each of the pattern detectors are associated with one of said defined coding patterns, including the step of the program detector manager invoking each of the pattern detectors, after one of the entry breakpoints associated with the coding pattern associated with said each of the pattern detectors, is reached in the computer program, for determining whether the computer program violates the coding pattern associated with said each of the pattern detectors.

Morshed, at the figures and related text cited to reject independent claims 1 and 15, does not include the novel debugging operational use of two levels of breakpoint insertions by a pattern detection monitor associated with a set of coding rules, to detect code patterns that violate same coding rules. That is, Morshed does not teach or suggest such a system that inserts entry breakpoints into the program, and during monitoring operation in which those entry breakpoints are detected, inserting at least one further breakpoint for identifying a respective step in the program that is part of the coding pattern associated with the entry breakpoints, and still less that

the plurality of defined coding patterns are selected from a group consisting of best practice patterns and problematic coding problems. Morshed does not seek to identify patterns based on rules.

Applicants respectfully assert that independent claims 1 and 15, as amended, are now patentably distinct from Morshed under Section 102. Claims 4-7 depend from amended independent claim 1 and are patentable therewith. Claims 17-19 depend from claim 15 and are patentable therewith. Applicants, therefore, respectfully request withdrawal of the rejections to claims 1, 4-7, 15 and 17-19 under Section 102 in view of Morshed.

With respect to the rejection claim 21, the Examiner asserts that Morshed discloses a method of detecting code patterns in a computer program that violates a given set of coding rules, the method comprising the steps of: defining a set of coding rules, each of the coding rules being associated with a respective one pattern detector (e.g., Fig. 14, blocks 446, 450, 454, 458, 462); providing a pattern detector manager for managing said pattern detectors (e.g., Fig. 14, blocks 442, 448, 452, 456, 460, 464, col. 23, line 1, to col. 24, line 11); providing a computer program, and running the computer program as a debug mode (e.g., Fig. 12, col. 21, lines 6-67 and col. 23, line 36, through col. 24, line 11); the pattern detector manager identifying, during the running of the computer program in the debug mode, points in the computer program that relate to said coding rules (e.g., Figs. 15-17, col. 24, line 11, through col. 25, line 67), and said pattern detector manager inserting into the computer program an entry breakpoint at each of said identified points (col. 20-lines 40-49, and col. 20, line 63 through col. 21, line 5); said pattern detector manager invoking each of the pattern detectors to monitor the computer program for a

violation of the coding rule associated with said each of the pattern detector (col. 21, lines 6-67), including the step of, each of the pattern detectors inserting one or more further breakpoints into the computer program to identify further points in the computer program that relate to the coding rule associated with said each of the pattern detector (col. 24, line 11, through col. 25, line 67), and tracking said additional breakpoints to determine whether the computer program violates the coding rule associated with said each of the pattern detectors (e.g., Fig. 14, col. 23, line 1, through col. 24, line 11, and col. 20, lines 6-62).

Applicants respectfully disagree. Applicants' independent claim 21 as amended sets forth a method of detecting code patterns in a computer program that violate a given set of coding rules. The method includes defining a set of coding rules, each coding rule of the set of coding rules being associated with a respective one pattern detector of a set of pattern detectors; providing a pattern detector manager for managing said pattern detectors; providing a computer program, and running the computer program in a debug mode; the pattern detector manager identifying, during the running of the computer program in the debug mode, points in the computer program that relate to said coding rules. The pattern detector manager inserts into the computer program an entry breakpoint at each of said identified points and invokes each of the pattern detectors to monitor the computer program for a violation of the coding rule associated with said each of the pattern detectors. The method further includes that each of the pattern detectors inserts one or more further breakpoints into the computer program to identify further points in the computer program that relate to the coding rule associated with said each of the pattern detectors and tracking said additional breakpoints to determine whether the computer program violates the coding rule associated with said each of the pattern detectors. Each of said

additional breakpoints identifies a respective step in the computer program that is part of the coding pattern associated with said one of the entry breakpoints, and wherein each of the pattern detectors monitors the computer program for the occurrence of any one of the first set of defined conditions, the occurrence of which violates the coding rule associated with said each of the pattern detectors and monitors the computer program for the non-occurrence of any one of a second set of defined conditions, the non-occurrence of which violates the coding rule associated with said each of the pattern detectors.

Morshed, at the figures and related text cited by the Examiner to reject independent claim 21, does not include the novel debugging operational use of two levels of breakpoint insertions by a pattern detection monitor associated with a set of coding rules, to detect code patterns that violate same coding rules. That is, Morshed does not teach or suggest such a system that inserts into the program a number of entry breakpoints, and during monitoring operation in which those entry breakpoints are detected, inserting at least one further breakpoint for identifying a respective step in the program that is part of the coding pattern associated with the entry breakpoints, and still less that the plurality of defined coding patterns are selected from a group consisting of best practice patterns and problematic coding problems. Morshed does not seek to identify patterns based on rules.

Applicants respectfully assert that independent claim 21, as amended, is patentably distinguishable from Morshed under Section 102. Claims 11 and 12, which depend therefrom Applicants, therefore, respectfully request withdrawal of the rejections to claims 21, 11 and 12 under Section 102 in view of Morshed.

If the Examiner believes that a telephone conference with Applicants' Attorneys would be advantageous to the disposition of this case, the Examiner is asked to telephone the undersigned.

Respectfully submitted,

A handwritten signature in black ink, appearing to read 'John F. Vodopia', written over a horizontal line.

John F. Vodopia
Registration No.: 36,299
Attorney for Applicants

Scully, Scott, Murphy & Presser, P.C.
400 Garden City Plaza – Suite 300
Garden City, New York 11530
(516) 742-4343

JFV:tb